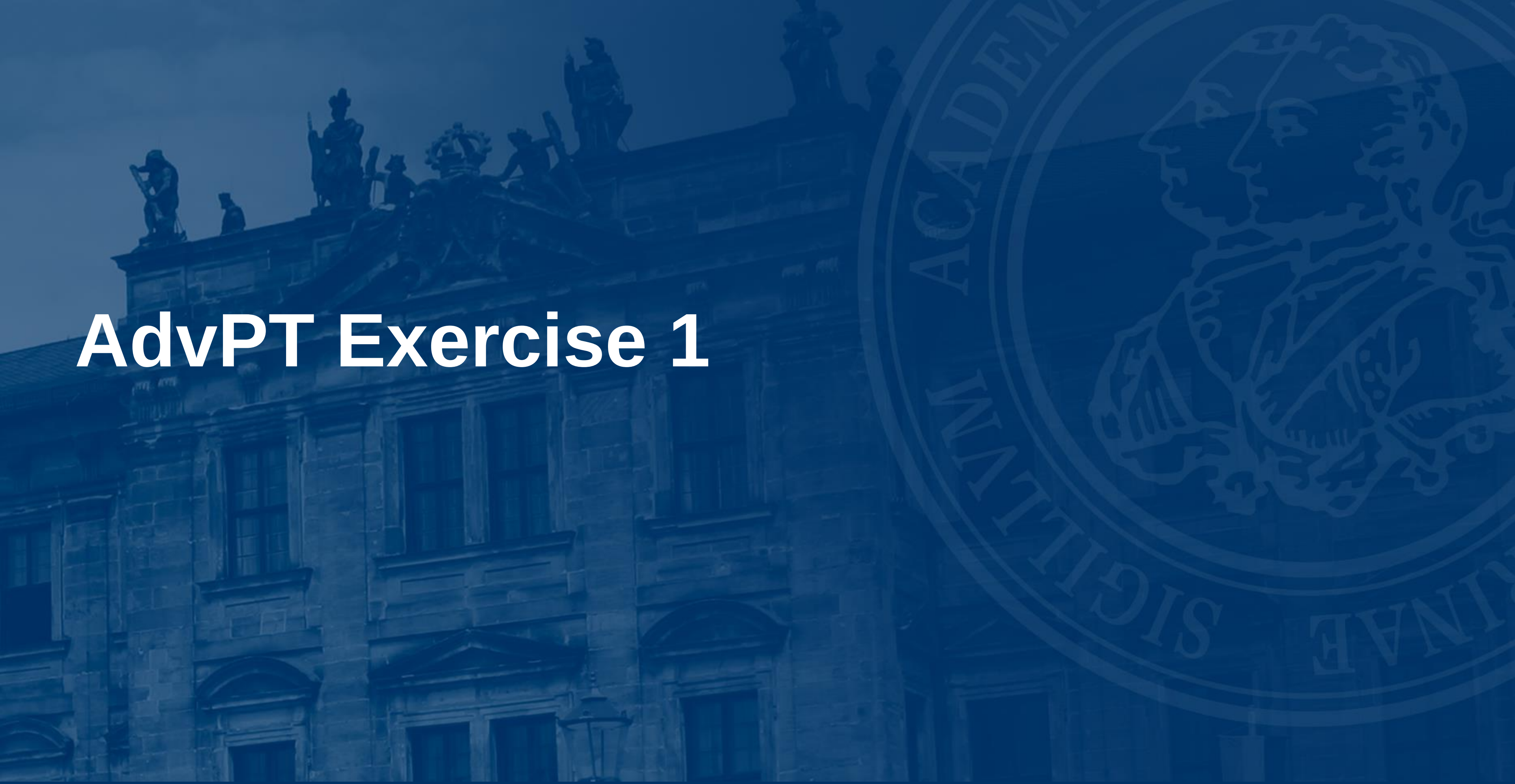


AdvPT Exercise 1

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Content

- Organization
- Development environment
- Memory management
- Assignment_A

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

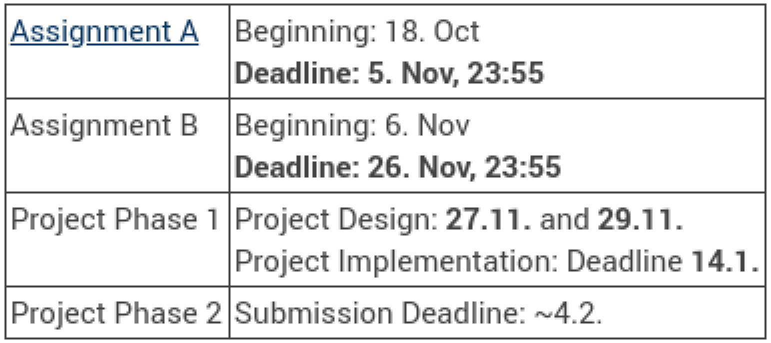
TECHNISCHE FAKULTÄT

Organization

- Time slot

Alternative time slot
Tuesday 12:30 - 14:00
Tuesday 14:00 - 15:30
Thursday 10:00 - 11:30
Thursday 12:30 - 14:00

Assignment plan

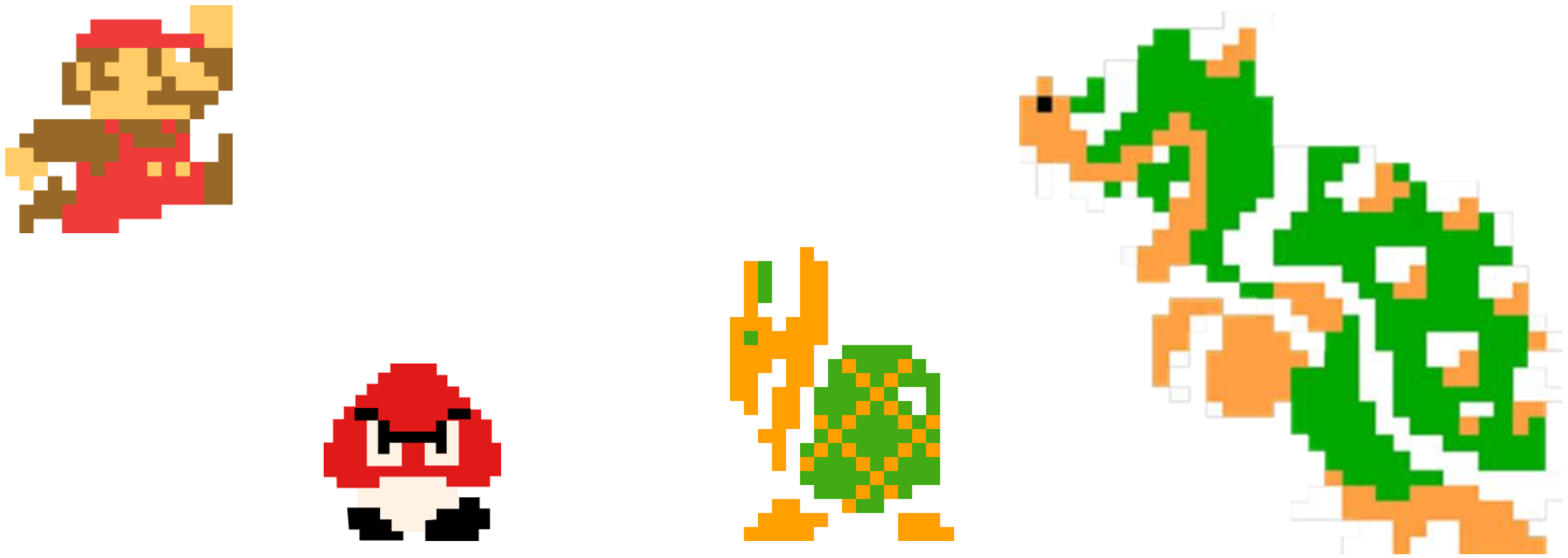


Assignment A	Beginning: 18. Oct **Deadline: 5. Nov, 23:55**
Assignment B	Beginning: 6. Nov **Deadline: 26. Nov, 23:55**
Project Phase 1	Project Design: **27.11.** and **29.11.** Project Implementation: Deadline **14.1.**
Project Phase 2	Submission Deadline: ~4.2.

Screenshot from Studon

Assignment plan

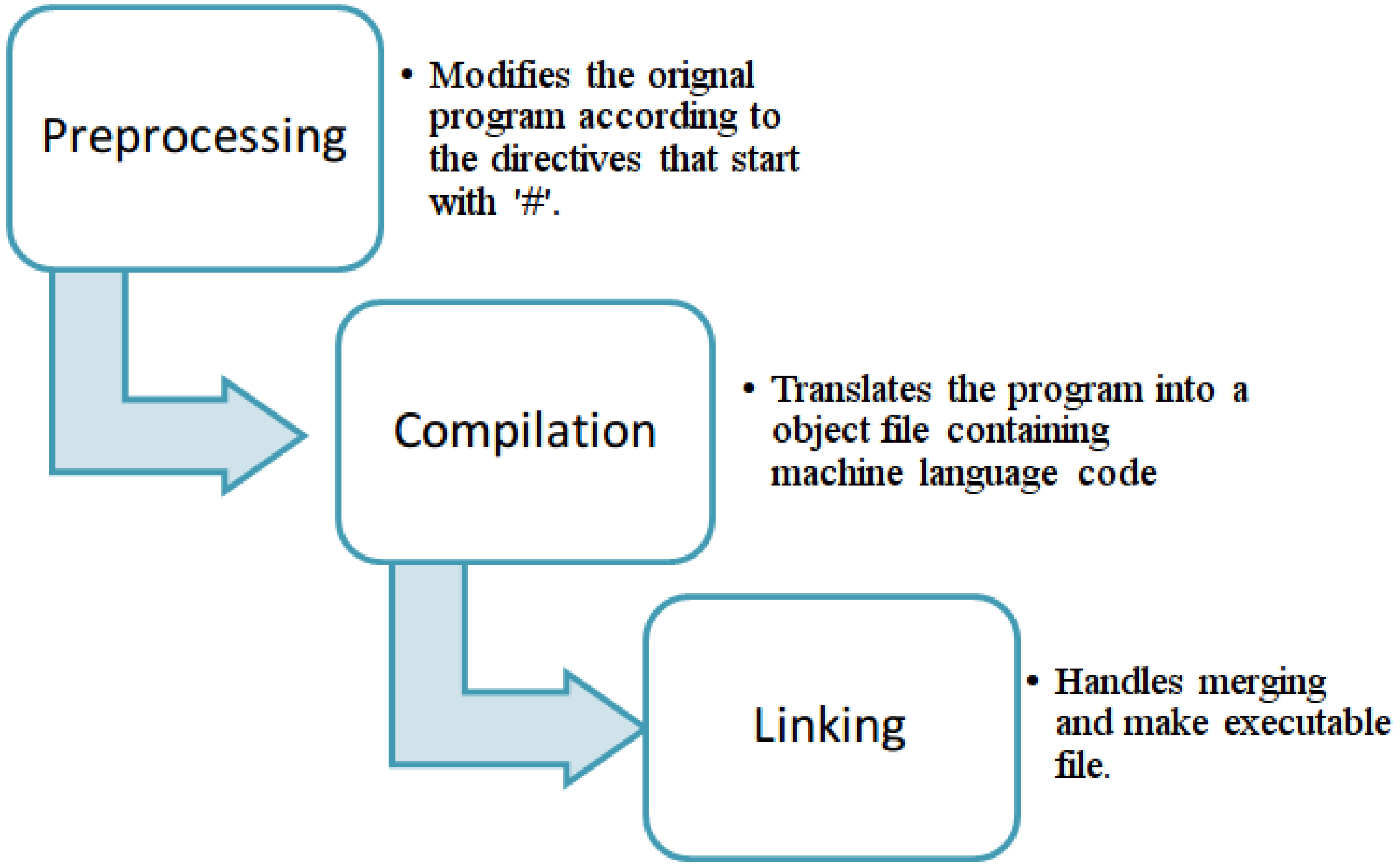
- Two Assignments and one project



Building a C++ project

- C++ compilers
  - **GNU Compiler**
  - **Clang (LLVM based)**
  - Intel Compiler
  - Microsoft C/C++ Compiler (Visual studio)
  - IBM Compiler
  - …

Building a C++ project



- Modifies the orignal program according to the directives that start with '#'.
- Translates the program into a object file containing machine language code
- Handles merging and make executable file.

Building a C++ project

- Compile Step:

```
g++  -c  MySourceFile.cpp -o MySourceFile.o
```

Flag	Meaning
-c	Compile
-g	Include debug information
-O3	Optimization Level 3 (Levels from 1 to 3)
-DVAR=1	Equivalent to “#define VAR 1”
-IMyIncludeDir	Add directory to search for includes

- Linking Step:

```
g++ MySourceFile.o MyOtherSourceFile.o -o myExecutable
```

-LMyLibraryDir	Add directory to search for libraries
-lgreatFeature	Links agains *libgreatFeature.a* or *libgreatfeature.so*

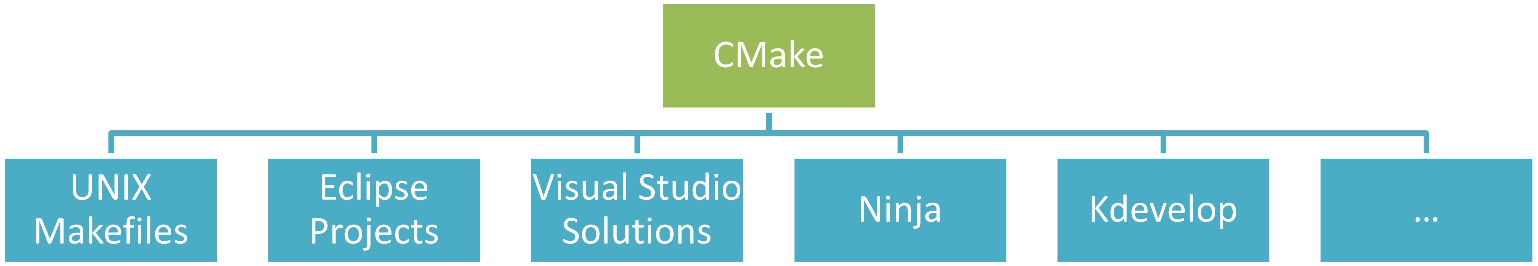
Building a C++ project

- Tasks of a build system
  - Search for available compiler
  - Handle dependencies - only rebuild parts that have changed
  - Assist with compile options (Debug/Release Mode)
  - Find libraries and generate according linking parameters
  - Handle installation/deployment
  - Integration of other tools (test drivers, documentation generators)
- Many build system available
  - autotools
  - Scons
  - Gradle
  - **CMake**
  - …

CMake

CMake because:

- Cross platform: generates build files for various infrastructures
- Well supported in modern IDEs
- Powerful (platform independent) scripting language



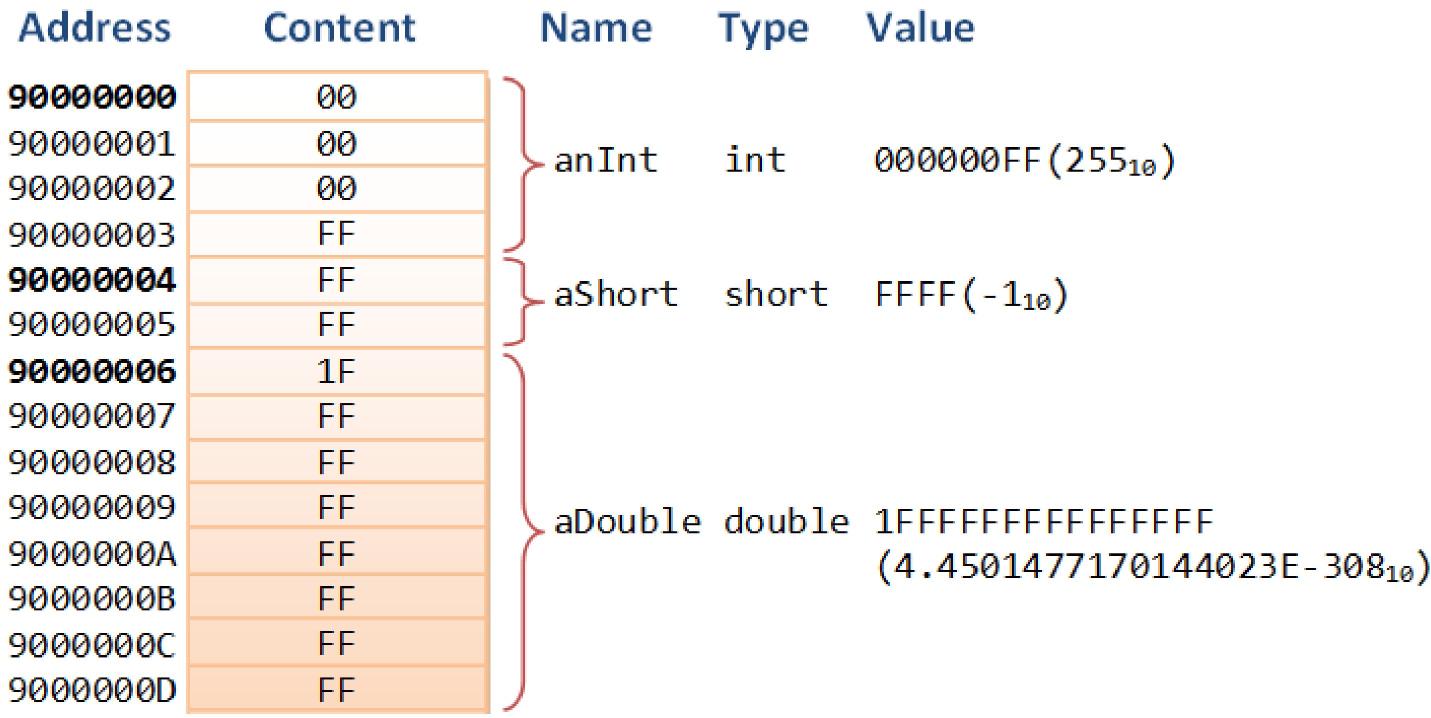
C++ IDEs

- QtCreator
- Emacs
- Vim
- Clion
- Eclipse
- Visual Studio

Memory management

- Memory management
  - Pointers & References
  - Multidimentional structures

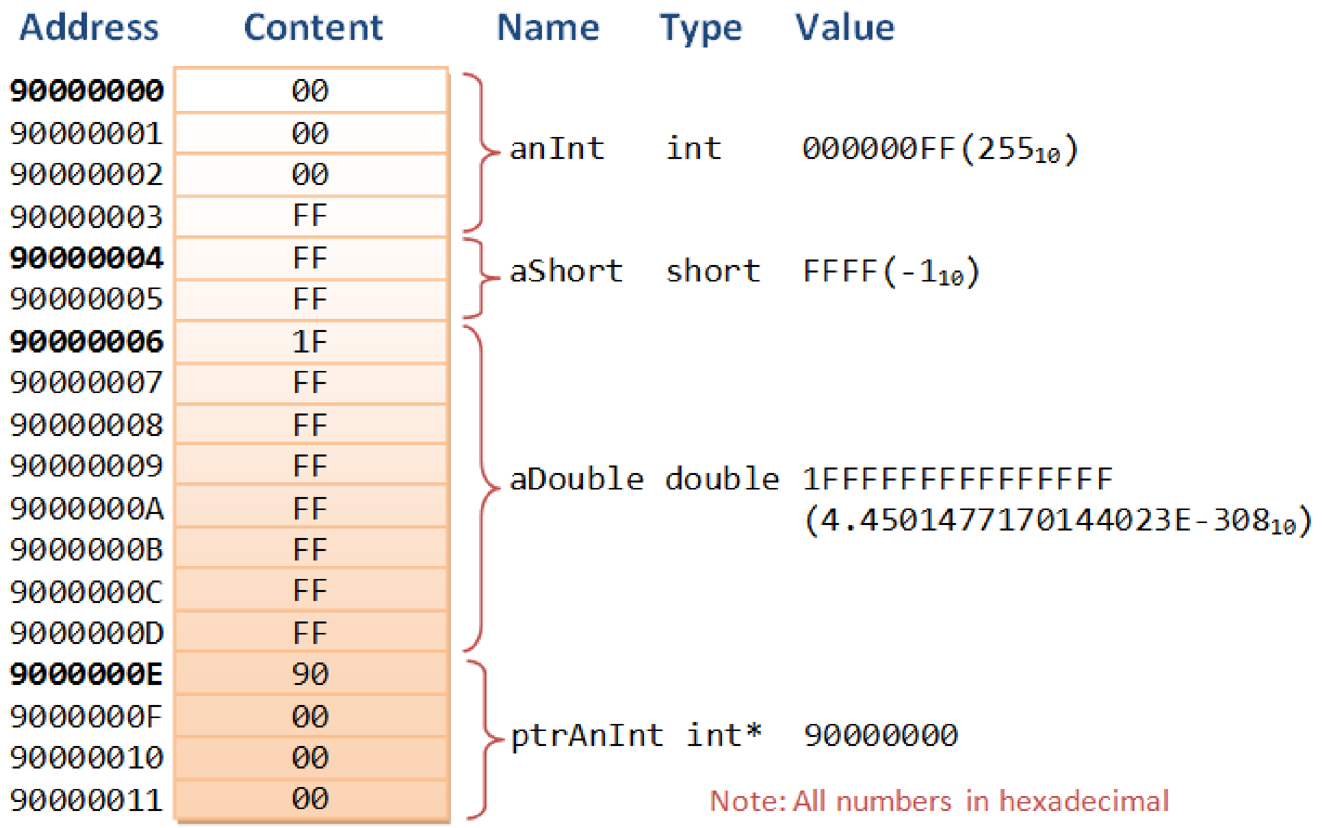
Pointer



Address	Content	Name	Type	Value
90000000	00	anInt	int	000000FF(255$_{10}$)
90000001	00			
90000002	00			
90000003	FF			
90000004	FF	aShort	short	FFFF(-1$_{10}$)
90000005	FF			
90000006	1F	aDouble	double	1FFFFFFFFFFFFFFF (4.4501477170144023E-308$_{10}$)
90000007	FF			
90000008	FF			
90000009	FF			
9000000A	FF			
9000000B	FF			
9000000C	FF			
9000000D	FF			

- memory: linear address space
- C/C++ gives you the power (responsibility!) to access arbitrary memory regions
- OS prevents access to certain memory regions (SEGFAULT)
- Pointer store memory addresses

Pointer

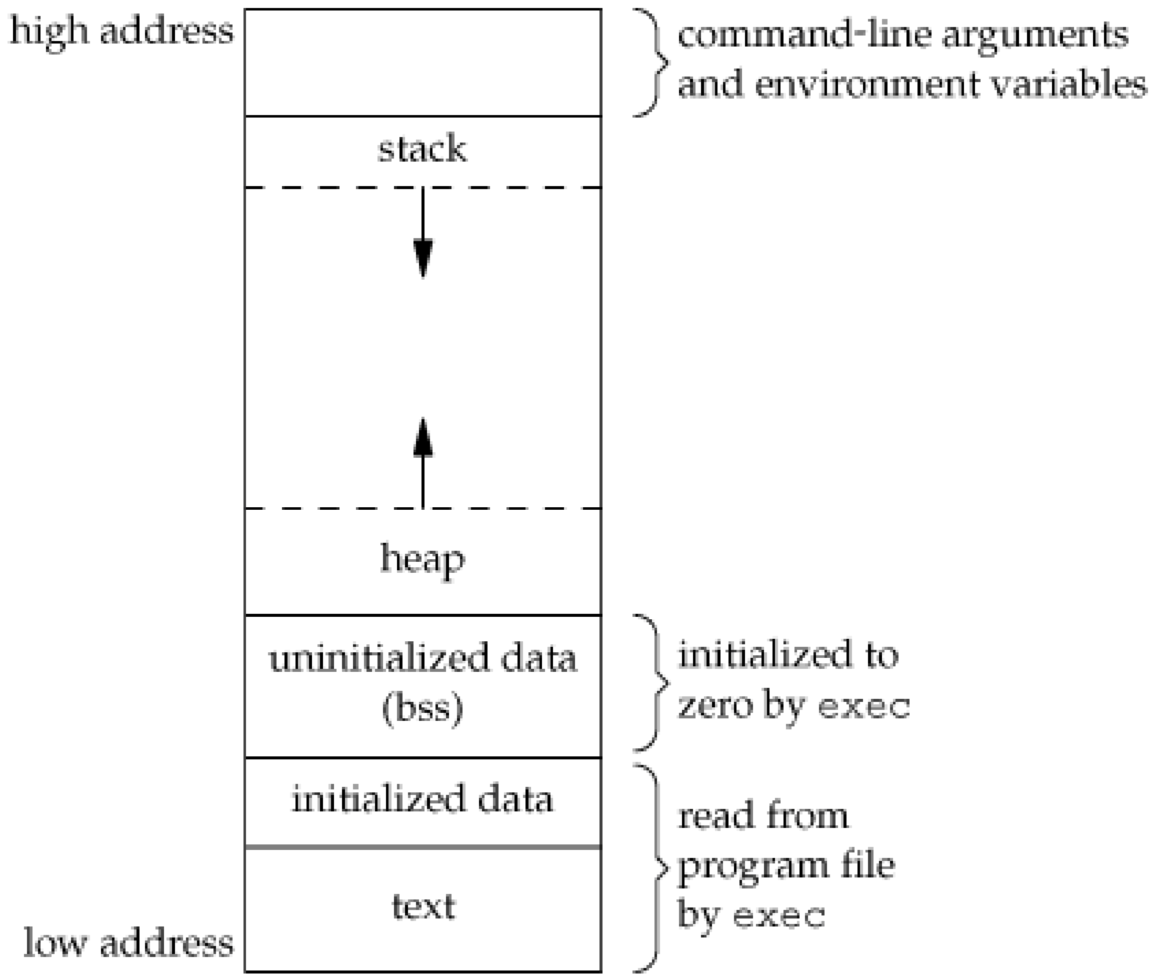


Address	Content	Name	Type	Value
90000000	00	anInt	int	000000FF(255$_{10}$)
90000001	00			
90000002	00			
90000003	FF			
90000004	FF	aShort	short	FFFF(-1$_{10}$)
90000005	FF			
90000006	1F	aDouble	double	1FFFFFFFFFFFFFFF (4.4501477170144023E-308$_{10}$)
90000007	FF			
90000008	FF			
90000009	FF			
9000000A	FF			
9000000B	FF			
9000000C	FF			
9000000D	FF			
9000000E	90	ptrAnInt	int*	90000000
9000000F	00			
90000010	00			
90000011	00			

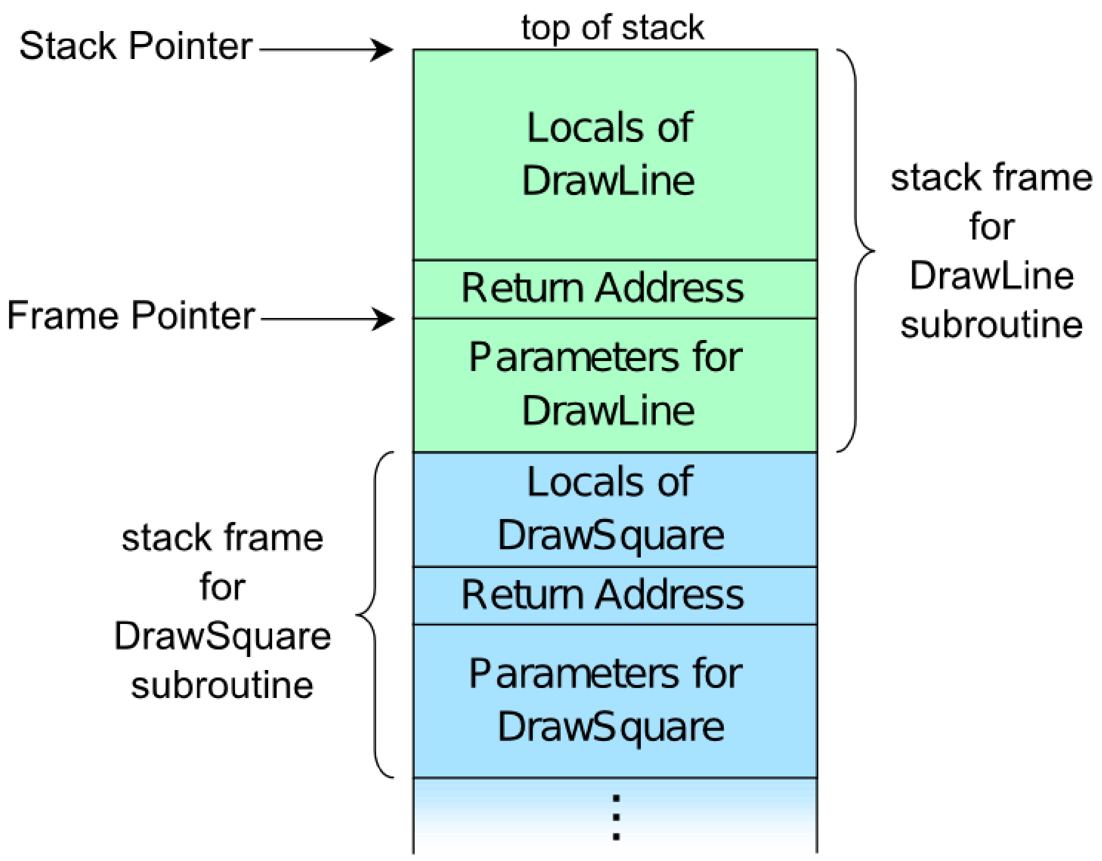
Note: All numbers in hexadecimal

- memory: linear address space
- C/C++ gives you the power (responsibility!) to access arbitrary memory regions
- OS prevents access to certain memory regions (SEGFAULT)
- Pointer store memory addresses

Memory region


high address
low address
stack
heap
uninitialized data (bss)
initialized data
text
command-line arguments and environment variables
initialized to zero by exec
read from program file by exec

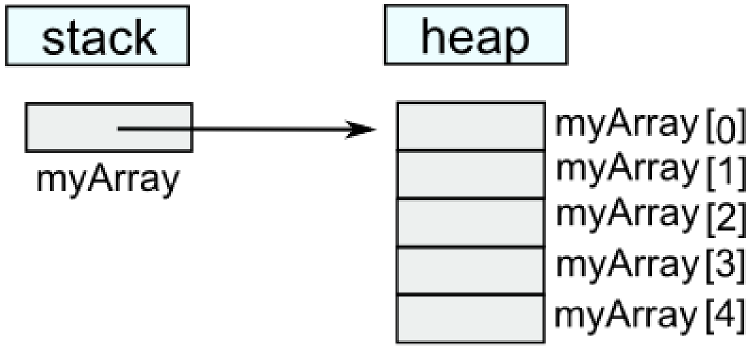
Stack



top of stack
Stack Pointer
Frame Pointer
Locals of DrawLine
Return Address
Parameters for DrawLine
Locals of DrawSquare
Return Address
Parameters for DrawSquare
stack frame for DrawLine subroutine
stack frame for DrawSquare subroutine

Heap

```
int * myArray = new int[5];
```



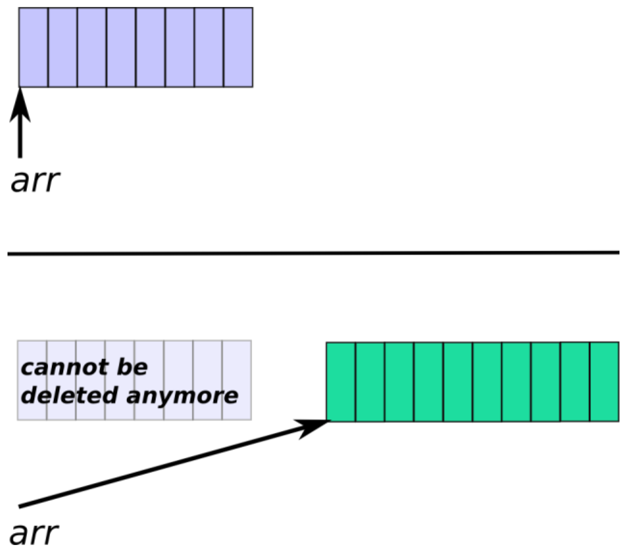
Memory Leaks

- Memory which is no longer needed is not released.

```
int main( int argc, char**argv)
{
    int * arr = 0;
    arr = new int[20]; //Leak!!

    return 0;
}


int main( int argc, char**argv)
{
    int * arr = 0;
    arr = new int[20];

    delete [] arr;

    return 0;
}
```

Common errors

```
int main( int argc, char**argv)
{
    int * arr = 0;
    arr = new int[20]; //Leak
    //do something with array, now larger array is needed
    arr = new int[40];

    delete [] arr;

    return 0;
}
```



Common errors

```
#include <iterator>
#include <iostream>

using namespace std;

int main( int argc, char**argv)
{
  int arr1[3] = { 1, 2, 100  };
  int arr2[2] = { 9, 8  };

  for( int i=0; i<=2; ++i )
    arr2[i] += arr1[i];

  copy( arr1, arr1+3, ostream_iterator<int>(cout, "," ) );
  cout << endl;
  copy( arr2, arr2+2, ostream_iterator<int>(cout, "," ) );
  cout << endl;
}
```

Common errors

```
int main( int argc, char** argv )
{
    int * arr = new int [500];

    for( int i=0; i<500; ++i )
        arr[i] = 0;

    delete arr;
}
```

Common errors

```
std::vector<int> & createUnitVector( int len )
{
    std::vector<int> vec ( 3, 1 );
    return vec;
}
int main( int argc, char** argv )
{
    auto myVec = createUnitVector( 4 );
    myVec[4] = 5;
    return 0;
}
```

Rule of three

- One special rule of thumb, which defines three special member functions:
  - Destructor
  - Copy constructor
  - Copy assignment operator

  The Rule of Three claims that **if one** of these had to be defined by the programmer, i.e. the compiler-generated version does not fit the needs of the class in one case and it will probably not fit in the other cases either.

- Example: Vector Class

Thank you and good luck!